2.0 Data Server

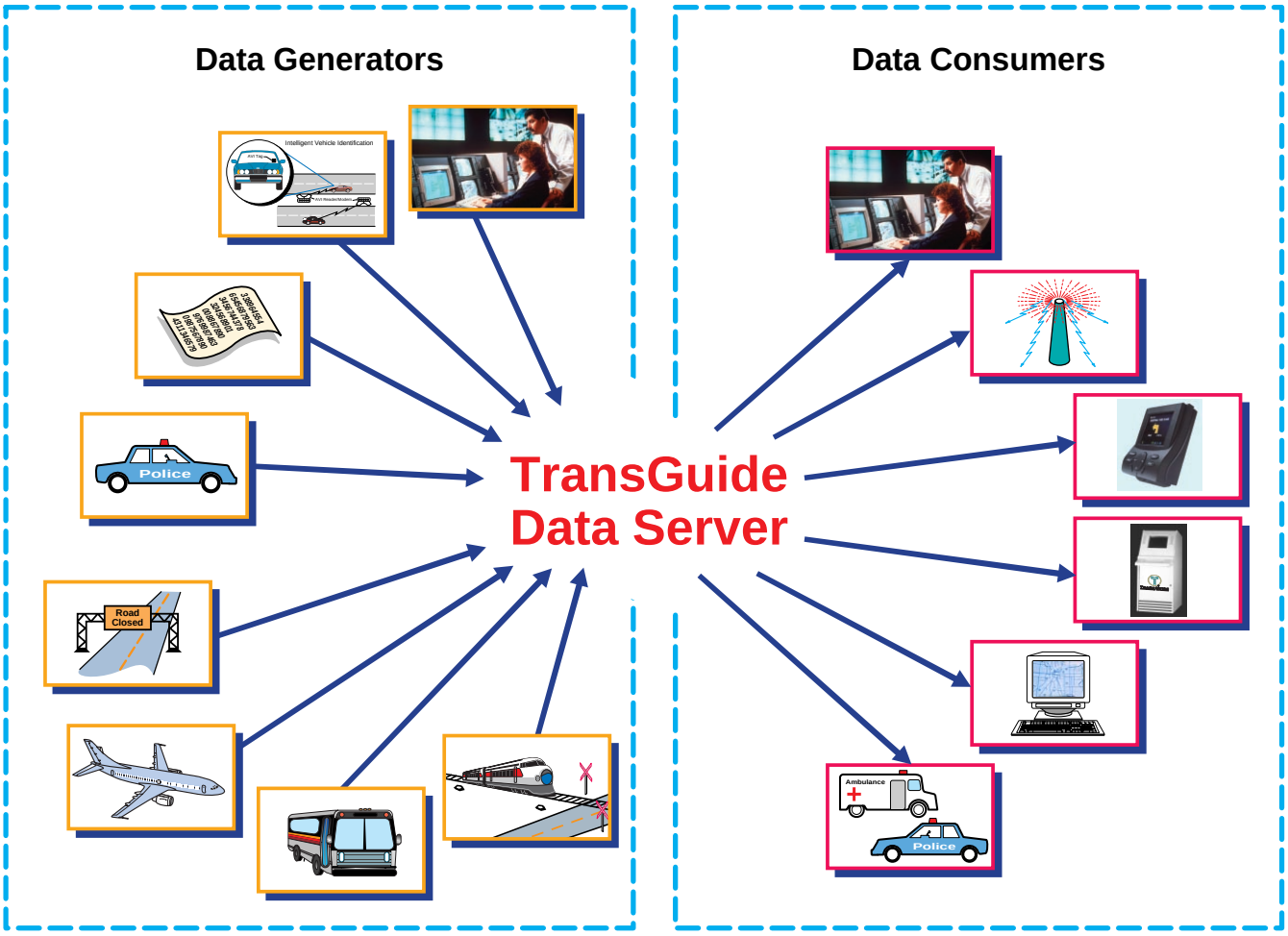

Figure 1. Data Server Conceptual Design

The Data Server is the central archive within the TransGuide environment for storing the data necessary to support the TransGuide ATMS operations and the MDI projects. Storing the data collected and utilized by the TransGuide system in a central common location allows the MDI projects to access the information they need seamlessly. The Data Server (illustrated in Figure 1) is developed around the concept of Data Generators, which supply data, and Data Consumers, which access the data.

The Data Server is developed around the concept of Data Generators, which supply data, and Data Consumers, which access the data.

2.1 Overview

The Data Server is not a traditional database. Instead, the data are stored in shared memory and in flat files. Data of limited and known size, such as speed data from the various types of TransGuide segments, are stored in shared memory. Data of varying sizes, such as ATMS incidents and lane closures, are stored in files. The Data Server

can also store files for later retrieval. In addition to traffic data, the Data Server receives a heartbeat message (i.e., a health and status message) from each of the MDI subsystems, and stores these data in shared memory.

Data Consumers can request the data and files sent to the Data Server by the Data Generators. The MDI System Status Graphical User Interface (GUI) accesses the heartbeat messages and other status information at regular intervals and displays them to the Traffic Operations Center personnel. This status GUI allows the user to determine the overall status of the MDI subsystems at a glance. Each subsystem has a display area on the GUI, which allows the user to get a detailed status of that subsystem.

2.2 Design Information

The Data Server is implemented by a number of processes executing on a Sun Microsystems Workstation. The following sections provide design details of the Data Server implementation.

2.2.1 Data Server Architecture

The Data Server design is based on a concept of Data Generators, which supply data to the system, and Data Consumers, which utilize these data. These Data Generators and Data Consumers can be internal or external to the system.

Some of the external Data Generators were developed specifically for the MDI project, while others were pre-existing applications. To avoid modifying the pre-existing applications, interface applications were developed for the external Data Generators.

To simplify the interface between the Data Server and the Data Generators or Data Consumers, a common library was developed that contains the functions needed to communicate with the Data Server. The library includes functions to send status information, traffic data, incident data, equipment data, and files to the Data Server. The library also includes functions to request traffic data, incident data, equipment data, and files or file-related information. By providing a common library to perform these tasks, system reliability and maintainability are improved. Any changes to the Data Server that affect the libraries need to be updated only once in the common library function, instead of in every application that communicates with the Data Server.

To increase the reliability of the Data Server, each individual Data Consumer, Data Generator, and external interface application is developed as a separate, independent module. This concept enables new Data Generators or Data Consumers to be added or existing ones to be updated with minimal impact. In addition, any failure in one module will probably not affect the others.

Figure 2 illustrates the Data Server data flow. The common library is shown in the figure as Application Program Interfaces (APIs).

2.2.2 Core Data Server Design

The core Data Server is the central module of the Data Server subsystem. It creates and maintains a common memory area, where it stores traffic data and status information about the various MDI systems. The Data Server also stores, retrieves, and

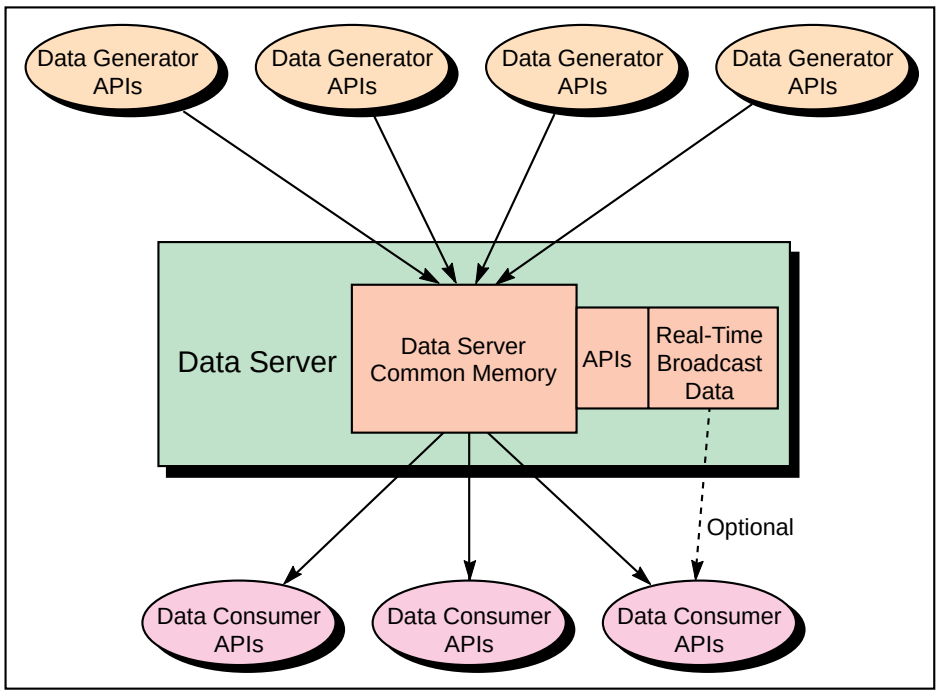

Figure 2. Data Server Data Flow

maintains information in files. Data of a known size, such as traffic speeds and system status, are stored in the common memory area. Traffic data of variable sizes, such as traffic incident information, are stored in files. Data Generators can also send files to be stored by the Data Server for future retrieval by Data Consumers. These files can be in any format because the Data Server does not access them other than to store them, retrieve them, retrieve directory information about them, or delete them.

Traffic congestion data are dynamic. Traffic incidents that occurred several hours ago are typically not pertinent to current traffic information. Therefore, the Data Server maintains only the most recent data. When the Data Server receives new speed data, it overwrites any previous speed value for the same road segment. When traffic incident data are received, the Data Server replaces any data from the same incident source (ATMS, 911, Lane Closures). In the case where the data source has failed to update the data within the expected time period, the Data Server uses an aging process to prevent old data from remaining in the system. For non-ATMS road segments, this process will set the status of the road segment to inactive and its speed to zero.

The core Data Server also starts and monitors the status of the Data Generators, Data Consumers, and interfaces that are part of the Data Server subsystem. This status is stored in a common memory area, which can be accessed by the MDI overall Status GUI (Figure 3). This Status GUI displays the current status of the MDI subsystems, allowing an user to determine at a glance the overall condition of the system. A color-coded status shows green when the subsystem has no errors, yellow when it is in a warning state, red when an error has occurred, and gray when the subsystem is not operating. The user can view the process status of each individual subsystem by selecting a button on the subsystem's status window. Some subsystems have detailed status windows that can be accessed from the subsystem's process status window.

The process status GUI for the Data Server subsystem is shown in Figure 4.

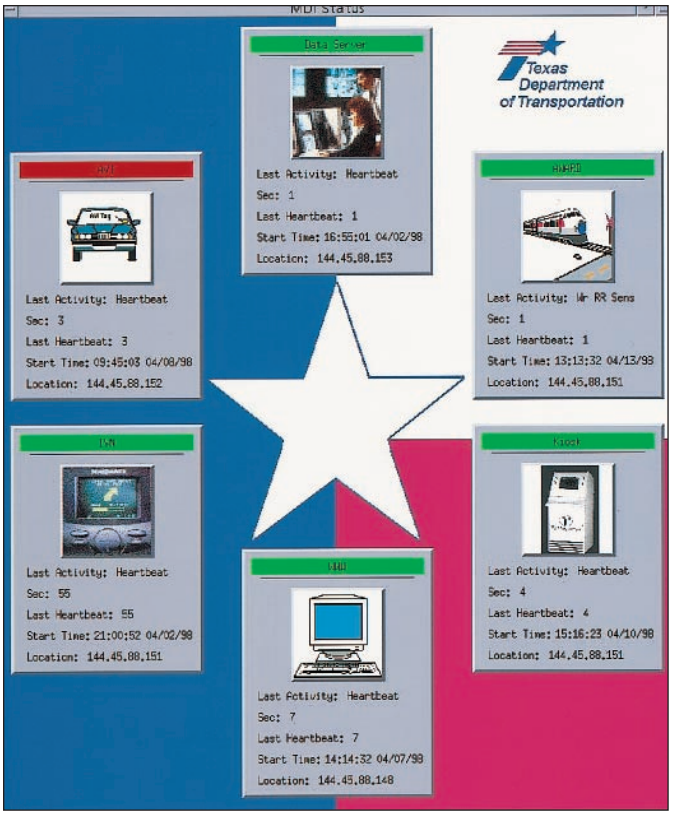

Figure 3. Overall Model Deployment Initiative Status Graphical User Interface

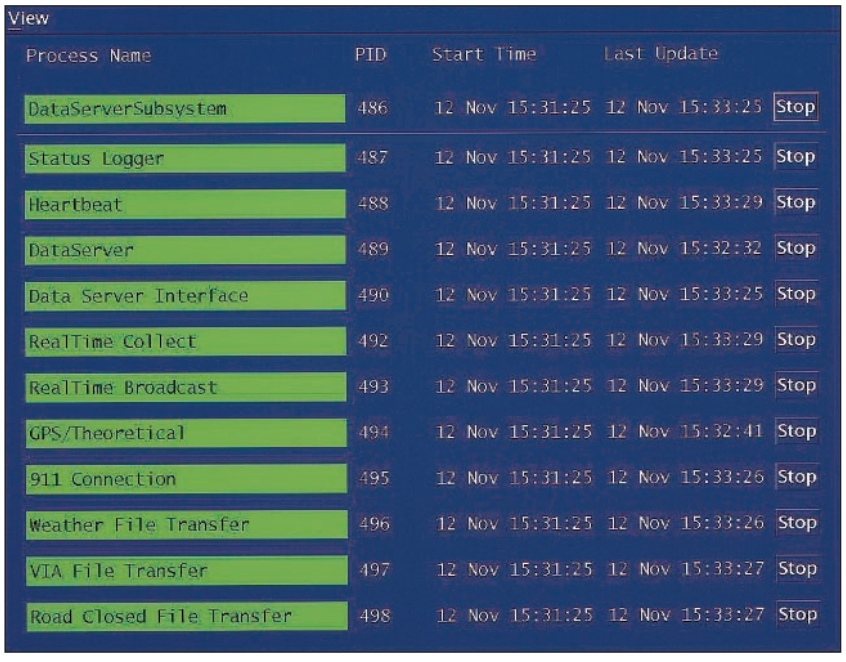

Figure 4. Data Server Process Status Graphical User Interface

2.2.3 Real-Time Subsystem Design

The Real-Time Subsystem (Figure 5) is the component of the Data Server system that interfaces with the real-time traffic data. These data are received from the ATMS by the Real-Time Collection application and stored in the Data Server common memory. The Real-Time Broadcast application retrieves these data in addition to AVI, Global Positioning System (GPS), 911, and Lane-Closure data from the common memory and broadcasts the data over the network. This broadcast can then be received by the Real-Time Receive application, which may run on one or more workstations connected to the network. An application, such as the Real-Time Map Display, can read the data stored by Real-Time Receive, and display it to a user wishing to see the current traffic speeds and traffic incidents.

2.2.4 911 Interface Design

The 911 interface receives traffic information from the San Antonio Police Department's 911 Data Dispatch and processes it before passing it on to the Data Server. The 911 data contains traffic accidents (minor and major) and other traffic-related incidents. Only accident-related information is sent to the Data Server. The 911 traffic accident data contain information about the accident such as type of accident, the time it occurred, and the geographic location. These data are converted into a standard MDI incident format and sent to the Data Server. The Real-Time Subsystem retrieves the incident information stored in the Data Server and broadcasts it on the TransGuide network. Other programs, such as the In-Vehicle Navigation (IVN) System, can use this incident information when making route guidance decisions. The geographic location information in the incident data also allows an incident icon to be displayed on the Geographical Information System (GIS) in the location at which it occurred.

2.2.5 Global Positioning System and Theoretical Data Subsystem Design

The GPS and Theoretical Data Subsystem is a Data Generator that supplies data to the Data Server. It retrieves traffic data from a database for road segments for which GPS and Theoretical data have been collected. GPS data are actual travel time data collected from GPS-equipped probe vehicles. Theoretical data are calculated using historical travel-time data combined with road characteristics. The database contains expected travel speeds for every 15-minute interval of each day of the week, for the selected road segments. The speed data are used to update the estimated travel speeds for the segments in the Data Server. Weather,

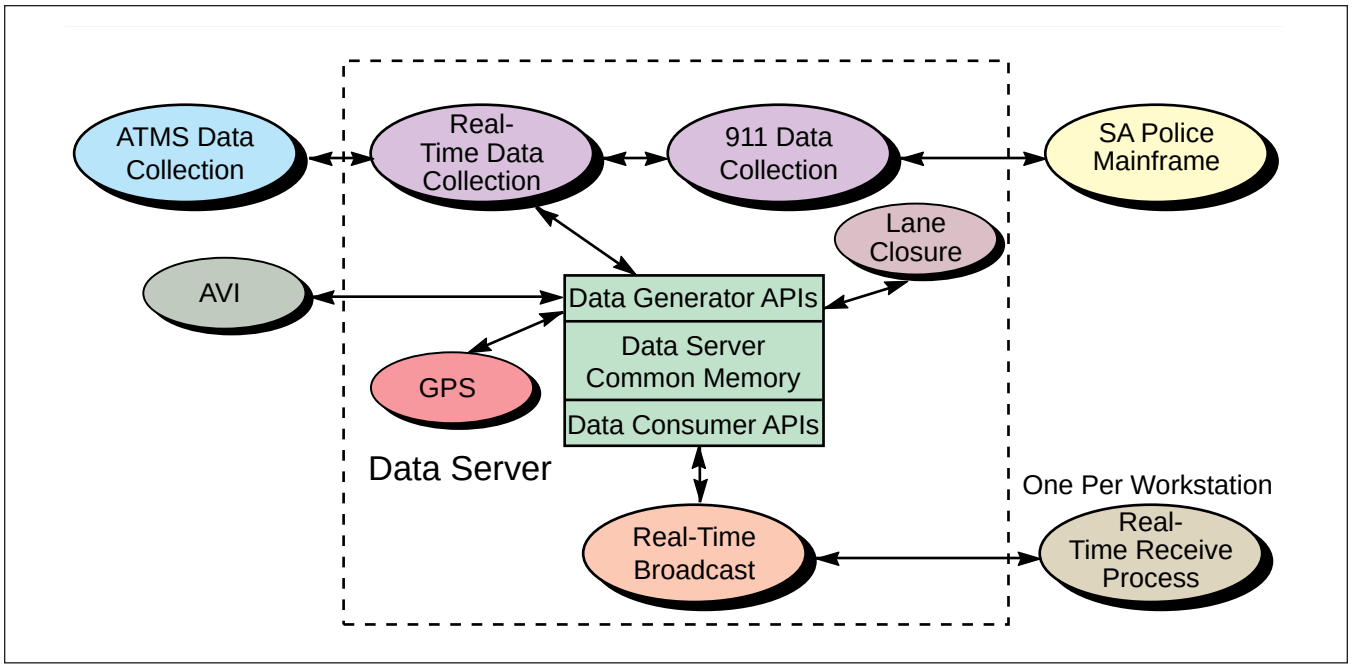

Figure 5. Real-Time Subsystem

school, and holiday information about the current day determines the values of a set of parameters used to adjust these speeds before they are transmitted to the Data Server.

Two GUIs were developed to facilitate entering the school, holiday, and weather attributes for individual dates or a range of dates. Using the Calendar GUI, school and holiday information can be entered up to a year in advance into a calendar file that is read each day at midnight to update the values for the current day. Another GUI allows the user to change these parameters for the current day. This GUI is mainly used to factor weather conditions into the traffic speeds for the GPS and Theoretical data segments.

2.2.6 File Transfer Design

The File Transfer process is a Data Generator that supplies data in the form of files to the Data Server. It is configurable to transfer weather files, VIA files, or highway condition files depending on the configuration values specified at startup. The source directory path, update frequency, and file type are among the configurable parameters.

At the specified intervals, the File Transfer process checks if any files in the source directory have been updated. The updated files are retrieved and sent to the Data Server where they are stored.

2.2.7 Real-Time Map Display Design

The Real-Time Map Display is based on the concept of intelligent map objects that contain geographic and attribute information for the object. The intelligent map objects are read in from a display file at startup. The Real-Time Map Display gets the real-time traffic information from the Real-Time Receive application. Real-Time Receive receives the data broadcast by Real-Time Broadcast onto the TransGuide network. These data are stored and can be requested asynchronously by applications such as the Real-Time Map Display. The traffic data contain speeds from ATMS segments, AVI segments, GPS segments, and Theoretical Data segments. ATMS incidents, 911 incidents, lane closure information, and MDI equipment information are also contained in the data feed.

The Real-Time Map Display draws the map based on the data in the display file. At regular intervals, the application reads the current speed, incident and equipment information and uses the data to update the map. The lane colors are determined based on the current speed and speed threshold settings—green is normal, yellow is warning, and red is alarm. The equipment icons and colors are determined by the current equipment status.

The user can obtain detailed speed, equipment, and incident information by clicking on a road segment or on an equipment or incident icon. This action causes a pop-up window to be displayed with a graphic representation of the selected lane's speed or detailed information about the status of the selected equipment or incident.

The user can also perform the following functions while viewing the Real-Time Map Display:

- Zoom the map display
- Scroll the map display
- Select one of two predefined views
- Show or hide lane closures
- Show or hide 911 incidents
- Show or hide MDI road segments

The Real-Time Map Display is also used to display the three map views on the video wall in the TOC. When operating in the video wall mode, the map display rotates between three predefined views that change at one-minute intervals.

2.2.8 World Wide Web Subsystem Design

The World Wide Web (WWW) subsystem runs on the TransGuide web server at the address http://www.transguide.dot.state.tx.us/map/. The server receives the Real-Time broadcast feed from the TransGuide network and uses these traffic data to update the road segments, lane closures, and traffic incidents that it displays. The WWW map displays real-time traffic speeds for ATMS and AVI instrumented road segments, traffic incident information about TransGuide ATMS incidents, and 911 Police incidents. The user can select from the following three map views:

- Real-Time traffic speed map
- Current traffic incident map
- Current lane closure map

The traffic incident and lane closure maps consist of the current traffic incidents or lane closures

indicated by yellow warning sign icons overlaid on the traffic speed map. A current traffic incident map is shown in Figure 6.

The traffic speed map shows the instrumented sections of road color coded to indicate the relative speed of that road section. A green color indicates that the traffic is flowing normally, yellow indicates that the traffic is slow, and red denotes very slow speeds. The cyan segments are currently inactive. When the user selects a section of road with the mouse, the current speed of that road segment is displayed above the map, along with a description of the location and direction of that segment.

When the traffic incident map or lane closure map is displayed, selecting one of the icons causes detailed information about the traffic incident or lane closure to be displayed. The user can also view a text list describing all current traffic incidents or lane closures.

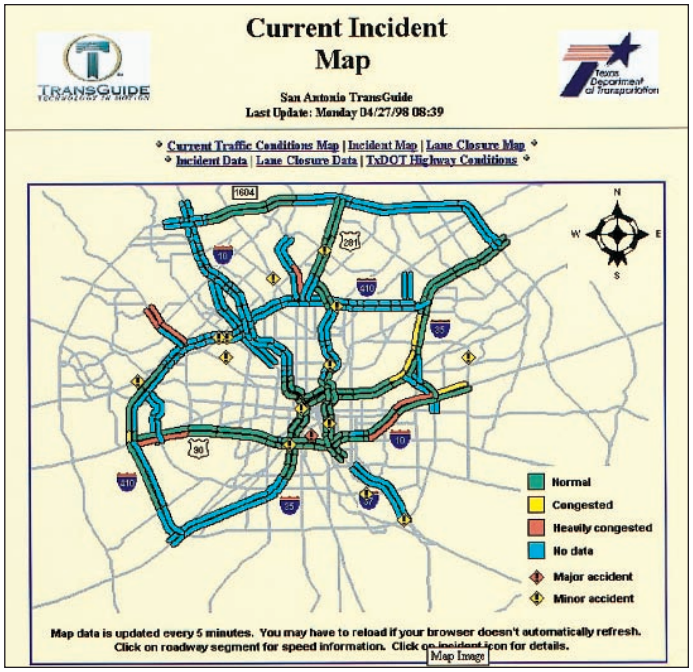

Figure 6. World Wide Web Current Incident Map

2.2.9 Data Server Common Library Design

To simplify the interface between other processes and the Data Server, a set of common Data Server Library functions was developed. Data Generators and Data Consumers use these functions to access Data Server information. The library consists of functions that allow the Data Generators to send traffic data, equipment data, and files to the Data Server and other functions that allow the Data Consumers to retrieve this data from the Data Server. Functions are also included to initialize and terminate communication with the Data Server, as well to send the Data Generator or Data Consumer's status to the Data Server.

2.2.10 Model Deployment Initiative Common Code Design

As design similarities were discovered in the various MDI projects, sets of common software libraries were developed and used across the program. These libraries, which became the groundwork for a common software architecture, provided several advantages during the initial MDI development, as it will during maintenance and future modifications and additions. Several of the common processes are customized using configuration data that are read from a file at start-up. The Data Server uses common libraries and code to accomplish the following functions:

- Data transmission to the Data Server and requests for data from the Data Server
- Heartbeat status collection and transmission
- Status logging
- Process start-up and monitoring

2.3 Tradeoff Decisions

Several tradeoff decisions were made during the design phase of the Data Server. Some decisions made on the other MDI subsystems also affected the Data Server design. Major tradeoffs considered for the Data Server included data distribution tradeoffs and architecture design tradeoffs.

2.3.1 Data Distribution and Process Breakup Tradeoffs

Several data distribution techniques were considered for the Data Server. Of these techniques, four were used for different types of data distribution. Each technique has different advantages as well as disadvantages, and each one was chosen for its particular application. The techniques are presented in Table 1.

Table 1. Data Distribution Techniques

	Broadcast	Shared Memory	Data Request	Files
Reliability/ Maintainability	High, relies on network.	Very High, requires careful access management (locks, semaphores, etc.). Does not rely on network.	Medium, easy to add new requests. Requires sender/ receiver cooperation. Relies on network.	Medium, relies on file system.
Speed	High, data can be compressed before transmission. No handshake protocol between client and server.	Very high.	Medium, two messages are sent for each request (the request and the response).	Medium, depends on disk access time.
System requirements	Low.	High, processes must reside on the same machine.	Low.	Low, file size adjusts to amount of data.
Expandability	High, sender is not impacted by an increase in number of listeners.	Medium, as number of accesses to shared memory increases, speed may slow due to waits.	Medium, a new process is spawned for each connection. Number of clients limited by system resources, network bandwidth.	High, limited by available disk space.

1. The data broadcast is used by the Real-Time subsystem to transmit real-time traffic data to a large (and variable) number of listeners located at several different workstations on a common network.
2. Shared memory is used by the Core Data Server to store data, which is accessed by two or more processes located on the same workstation. An example of these data is the status data collected by the Data Server process and read by the MDI overall Status GUI. Traffic data with a known number of elements are also stored in shared memory.
3. Data Requests are used by other MDI subsystems to send data to, or request data from, the Data Server. These processes may or may not be located on the same workstation as the Data Server.
4. Traffic data with a varying number of data elements, such as ATMS incidents, 911 incidents and lane closures, are stored in files. As the number of elements change, the size of the file changes, without requiring reallocation or waste of memory.

The tradeoff matrix for the different data distribution techniques is shown in Table 1.

2.3.2 Architecture Design Tradeoffs

The Data Server consists of several subsystems and interface processes. Instead of creating one large application program to implement all the subsystems and interfaces, the subsystems and interfaces were split into several programs. Each subsystem consists of one or more component, each of which is a separate UNIX process. The subsys-

tem components were designed as individual processes to increase the reliability of the system. Hence, if a system has to be modified or added, or if one process fails, it will not impact the whole system.

The tradeoff matrix for the architecture design is shown in Table 2.

2.4 Summary

The Data Server is designed to provide a seamless integration with the existing TransGuide operational environment. The Data Server provides rapid access to the various types of traffic data that it stores. Because of its modular design, the Data Server is flexible, making future additions of Data Generators or Data Consumers possible.

The Data Server project collects traffic data from multiple sources and distributes the data to external applications, such as Kiosks, IVN, and the Real-Time WWW map. Traffic data sources include speed data from ATMS sensors, AVI sensors, and GPS/Theoretical computations; traffic incident data from the ATMS, 911 police data dispatch system, and AWARD sensors; equipment data from ATMS equipment and MDI equipment, as well as scheduled lane closures. The traffic data are presented in an easily readable Real-Time map with color coded speed ranges. This map can be zoomed to obtain various levels of detail without sacrificing geographical accuracy.

The Data Server implementation has shown that adding and removing applications that provide and use the traffic data can be done with minimal effort. Additional roadways and equipment are currently being added to the system by using an intuitive map generation application developed in parallel with the Real-Time map. This process shows that the system can be updated to keep up with continuing development of the San Antonio traffic system.

Table 2. Data Server Architecture Tradeoff Matrix
(Selected technique is highlighted in yellow)

	One Process	Separate Processes
Maintainability	Medium.	High.
Reliability	Low, if an error occurs, the entire system goes down.	High, errors affect only the subsystem at which they occur.
Flexibility	Low.	High, change in one process does not necessarily affect other processes.
Expandability	Medium.	High, easy to add or delete processes.